LPCXpresso 1769 CPU Module IrDA and ExINT with LCD Testing

Troy Kurniawan
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 95192
E-mail: troy.kurniawan@sjsu.edu

Abstract

This report discusses a lab that creates a system using an LPCXpresso 1769 CPU module to realize an external interrupt signal that interacts with an LCD screen. The main challenges were to understand the infrared (IR) receiver and the LCD screen's physical capabilities and how to write code to use the systems. The lab was successful in creating an external interrupt system that interacts with an LCD screen.

1. Introduction

In this lab, a prototype microprocessor board was designed using the LPCXpresso 1769 CPU [1]. This board is a CPU module that has many capabilities that can be programed through a computer using a software IDE called "MCUXpresso." For this lab, different objectives were called for in the form of various mini circuits. The board contains a programmable liquid-crystal display (LCD) screen [2] as well as a TSOP532 IR receiver [3]. Furthermore, C code was written to program the board and its functionality. Much of the theory work was provided by the professor, which helped guide this lab towards its success [4]. However, other challenges were also present, such as building the physical border using a soldering kit. Outside resources were available to help out with the lab.

An LCD screen is an electronic display that is able to produce images in various colors. To test the functionality of the LCD screen, a program was written to test its ability to draw various shapes and figures. For this lab, code was written to draw a series of rotating squares that spirals inwards.

To further expand the board's capabilities, an IR receiver was added to interact with the LCD screen. An IR receiver is a device that is able to detect IR signals from an external device and output a signal. For this lab, code was written that would draw green and red squares on the LCD screen whenever an IR signal was sensed by the receiver.

This report is split into a variety of sections to cover the project's methodology, hardware and software implementation, and conclusion.

2. Methodology

This methodology section is split into two sections which include the "Objectives and Technical Challenges" section and the "Problem Formulation and Design" section. The first section presets the various goals the lab aims to achieve, and the second section tells of the approach to designing the board and code.

2.1. Objectives and Technical Challenges

The main task for this lab was to design a functional prototype board with a CPU module. Three other objectives were also presented. The first was to fully connect an LCD screen and test it by using C code to draw various shapes. The screen has various pins that need to be taken into account. Careful understanding of the pins is necessary.

The second objective was to use an IR receiver to control images that displayed on the LCD screen. It is imperative to be aware of the capabilities of the IR receiver and how the pins interact with the CPU module.

The third task was to write code using the MCUXpresso IDE so that the CPU module can receive signals from the IR receiver and control the LCD screen. Certain special purpose registers (SPRs) are required for this part. In addition, an algorithm needs to be written in order to display rotating squares

Various other technical tasks are also presented. Soldering and wire wrapping are two prevalent challenges that appeared for this lab.

2.2. Problem Formulation and Design

It was a challenge to design the layout of the board and its modules in an effective way. Unfortunately, space was limited. As a result, the LCD screen hangs off the top edge of the board. This is certainly not the ideal location for it, but it was the only spot that could fit it. Various pin connections were made between the Fortunately, the IR receiver is very small and could fit in most spots on the board. It was placed at the edge because it was the most open spot to receive IR signals. In addition, the location of the IR receiver was close to the necessary pins on the LCP module, which made connections more convenient. The full layout can be found in figures 1 and 2.

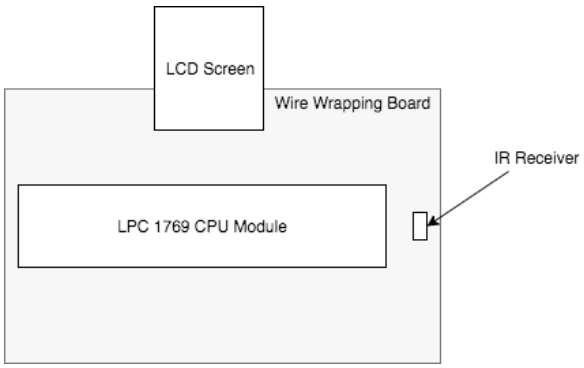

Figure 1. The top-level drawn out layout view of the prototype board.

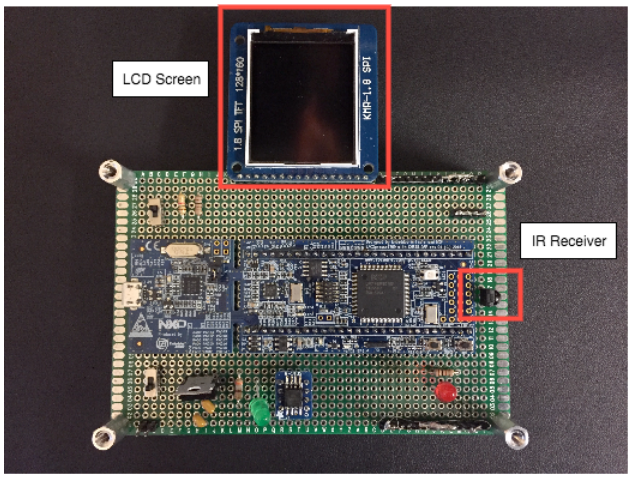

Figure 2. The top-level physical layout view of the prototype board.

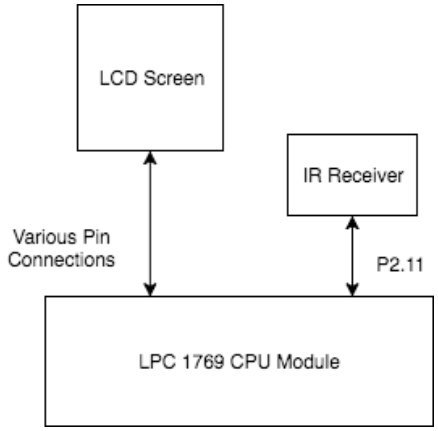

Figure 3. The system block diagram of the LCD Screen and the IR receiver connected to the LPC1769 module.

The LCD screen has various pins. The "LED-" and "GND" pins serve as the ground for the screen. The "LED+" and "Vcc" pins serve as the main power source for the screen. The "MISO" (Master Input, Slave Output) pin serves as a way for the screen to send data back to the CPU module. The "CS" (Chip Select) pin that tells the screen to receive data from the CPU module as opposed to an SD card input. The "SCL" (Serial Clock) pin serves as the clock input for the screen. The "SDA" pin serves as the data input where the CPU module can send data to the screen. The "AO" pin serves as a data or command selector pin. The "RESET" pin serves as a way to fully reset the screen.

The IR receiver has three pins. Pin one is the "out" pin, which sends an electrical signal out to its destination when an infrared signal is received. Pin two is the "ground" pin which serves as the module's ground. Pin three is the "Vcc" pin which serves as the main power source for the module. More details about the hardware can be found in the hardware section of this report.

The first part of this lab tested the capabilities of the LCD screen by drawing a series of rotating squares. Drawing a square on the screen is done by utilizing the "drawline" function provided to us by the professor. This function draws a simple line given two sets of coordinates: a starting coordinate and an ending coordinate. The function contains five arguments: two x-axis coordinates, two y-axis coordinate, and the color of the line (represented by a six-digit hexadecimal value). Applying four "drawline" functions with the specific coordinates can draw a square on the screen. To draw a series of rotating squares, new x and y coordinates need to be calculated so that every rotated square is changed proportionally to each other. To do this, an equation is needed to change the values of these x and y coordinates:

$$\overrightarrow{P}_i^{j+1} = \overrightarrow{P}_i^{j} + \lambda(\overrightarrow{P}_{i+1}^{j} - \overrightarrow{P}_i^{j})$$

This equation calculates the next vector (or the x and y coordinates of a line) given the previous one. The variable "j" represents which rotated square is being calculated (also known as, the "layer" at which the square is being drawn on). The variable "i" represents a single point on the square. The variable lambda "λ" represents a way of taking a fraction of the distance of the current line to prepare for the next line. For example, selecting "0" for lambda would result in giving the starting point of the line. Selecting "1" for lambda would result in giving the ending point of the line. Choosing a number between one and zero would result in a point somewhere between the start and end points. The equation fully states that the point "i" on the next layer's square is determined by the same point on the current square added to a fraction of the distance between the current line's start and end points. Further application of this equation and the code can be found in the implementation section of this report. Pseudocode can be found in the software section of this report.

The second part of the lab tested the interaction between the IR receiver and the LCD screen. When the IR receiver detects a signal, a red square would appear on the LCD screen. When another signal is detected, the square would turn green. Every signal detected afterwards would simply result in an alternating sequence between red and green squares. For this lab, a standard TV remote controller was used to send infrared signals to the receiver.

3. Implementation

This implementation section is organized into two sections which include the hardware's design and the software's design. The hardware's design was based off of a schematic provided by the professor. Additions and edits were made to that original template. The software's design was also based on previous work done by the professor. Again, changes were made to the original work to reflect the intended design.

3.1. Hardware Design

The LCD screen's pins were connected to various parts of the CPU module. Both of the "GND" and "LED-" pins on the screen were connected to the J2-1 pin of the CPU module, which is the ground pin. Both of the "Vcc" and "LED+" pins on the screen were connected to the J2-28 pin of the CPU module, which is the 3.3V pin. The "MISO" pin is connected to the "MISO" pin P0.17 on the CPU module. The "CS" pin is connected to the "CS" pin P0.16 on the CPU module. The "SCL" pin is connected to the "SCK" pin P0.15 on the CPU module. The "SDA" pin is connected to the "MOSI" pin P0.18 on the CPU module. The "AO" pin is connected to the "GPIO-DC" pin P0.21 on the CPU module. The "RESET" pin was connected to the "GPIO-RESET" pin P0.22 on the CPU module. Its schematic can be found in figure 4.

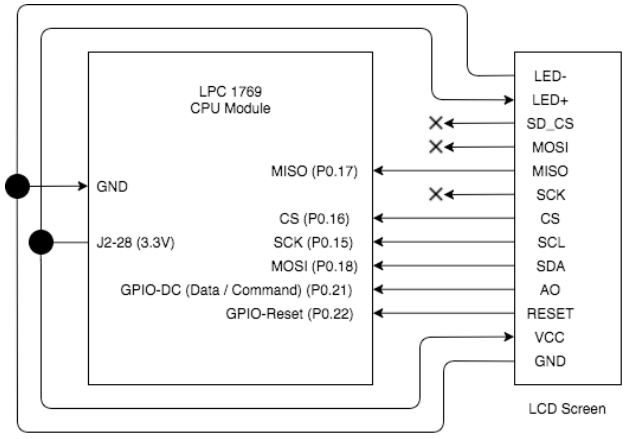

Figure 4. The schematic of the power circuit connected to the LPC1769 module.

The IR receiver has three simple connections. Pin one of the receiver is the "out" pin and goes to the external interrupt input P2.11 on the CPU module. Pin two is the "GND" pin and goes to the J2-1 ground pin of the CPU module. Pin three is the "Vcc" pin and goes to the J2-28 3.3V power supply pin on the CPU module. Its schematic can be found in figures 5 and 6.

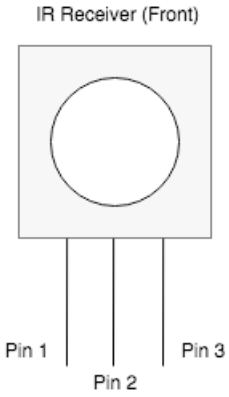

Figure 5. The schematic of the IR receiver.

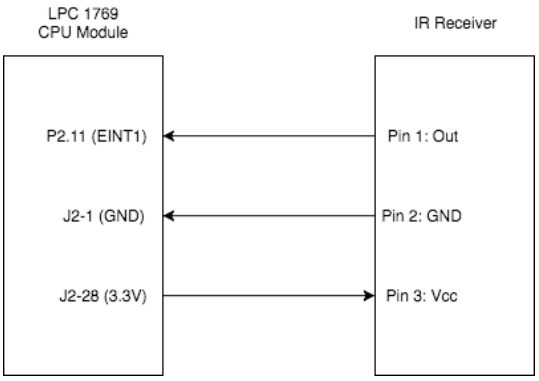

Figure 6. The schematic of the IR receiver circuit connected to the LPC1769 module.

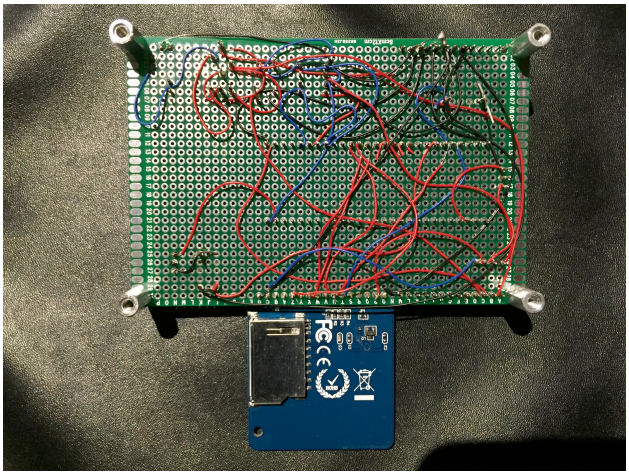

Figure 7. The under side of the board, showing the various connections.

Bill of Material (List of Components):

- PCB Board
- 1/2in. Standoffs
- LPC1769 Xpresso CPU Module
- 30AWG Wires
- 1.8 SPI TFT 128*160 LCD Screen
- TSOP532 IR Receiver
- Solder

The full kit containing all the parts totaled to $75.

Table 1. Connection Table that lists the different pins on the CPU module and what it's connected to on the board.

LPC1769 Pin:	**Connection:**
J2-1 (Ground)	LCD Screen's LED- and GND IR Receiver's GND
P0.15 (SCK)	LCD Screen's SCL
P0.16 (CS)	LCD Screen's CS
P0.17 (MISO)	LCD Screen's MISO
P0.18 (MOSI)	LCD Screen's SDA
P0.21 (GPIO-DC)	LCD Screen's AO
P0.22 (GPIO-RESET)	LCD Screen's RESET
J2-28 (3.3V Output)	LCD Screen's LED+ and Vcc IR Receiver's Vcc
P2.11 (EINT1)	IR Receiver's Out

3.2. Software Design

The code for drawing a set of rotating squares on the LCD screen first begins by initializing the LCD screen using the "lcd_init()" function, which uses the SPRs FIODIR, FIOSET, and FIOCLR to establish pin assignments for the LCD screen and CPU module. The code then proceeds to fill the screen with a black background. For aesthetic purposes, a white board was also drawn at the edges of the screen. Four pairs of x-axis and y-axis variables were declared to be used for the four "drawline()" functions that are used for drawing each layer's square. A for loop is then used to iterate the drawing code five times. The loop begins by using the declared and initialized variables to draw the first layer's box. The loop then proceeds to use the equation mentioned previously to calculate new values for each variable. The equation is used eight times per loop (one equation for each variable). The lambda variable was set to "0.2". Once the loop is finished, the code is complete and exits.

Algorithmic Procedure:

1. Initialize the LCD screen.
2. Fill the screen with a black background and a white border.
3. Declare and initialize four pairs of x-axis and y-axis variables.
4. Draw square using the variables.
5. Use the equation to calculate new x-axis and y-axis values for the next layer's square and repeat.

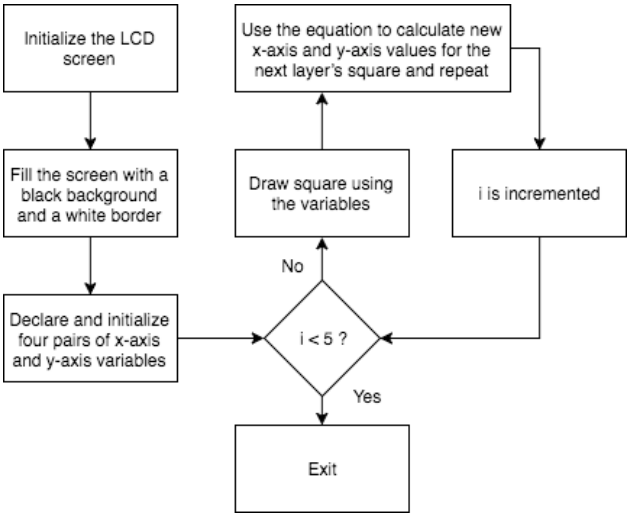

Figure 8. The C code main function flowchart for the LCD screen rotating squares test.

LCD Screen Pseudocode:

```
Initialize LCD screen;

Declare and initialize x-axis and y-axis variables;

for(int i=0; i<5; i++){

        Draw square using the variables.

        Use equation to calculate new values for the
        variables

}
```

The code for the IR receiver and LCD screen interaction begins by establishing pin connections between the CPU module, IR receiver, and LCD screen. Using the SPR FIODIR, pin P2.11 is used as the input signal for the IR receiver. Like before, the LCD screen is initialized and a black background is displayed. An infinite while loop is then written to call the "EINTInit()" function, which checks a signal from the IR receiver consistently. Within the function, the SPR EXTINT and FIOPIN is used to receive an interrupt signal at the specified pin (P2.11). If a signal is received, the code enters an if statement. Two blocks of code are found here. One block of code contains "drawline()" functions that draw a red square. The other does the same but for a green square. The code is written so that every time a

signal is received, the code enters each block of drawing code individually and alternately. The code ends by clearing the interrupt pins.

Algorithmic Procedure:

1. Initialize the LCD screen.
2. Enter a while loop to check pin P2.11 for an interrupt signal.
3. If a signal is received, draw a red or green square.
4. Repeat steps two and three.

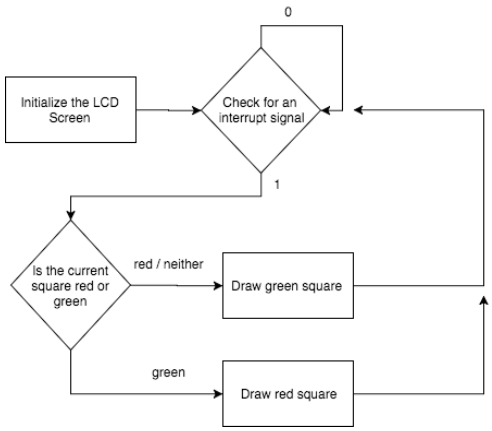

Figure 9. The C code main function flowchart for the IR receiver and LCD screen circuit.

LCD Screen and IR Receiver Pseudocode:

```
Initialize LCD screen

while(1){
        if(interrupt signal received){
                if(current square is red){
                        Draw green square
                }
                if(current square is green){
                        Draw red square
                }
        }
}
```

Source code for all circuits can be found in the Appendix section of this report.

4. Testing and Verification

The LCD screen rotating square test ran successfully. As the code ran, the LCD screen initialized by clearing the screen and swiping in a black background. The rotating squares appear in blue one at a time. The LCD's image can be found in figure 11.

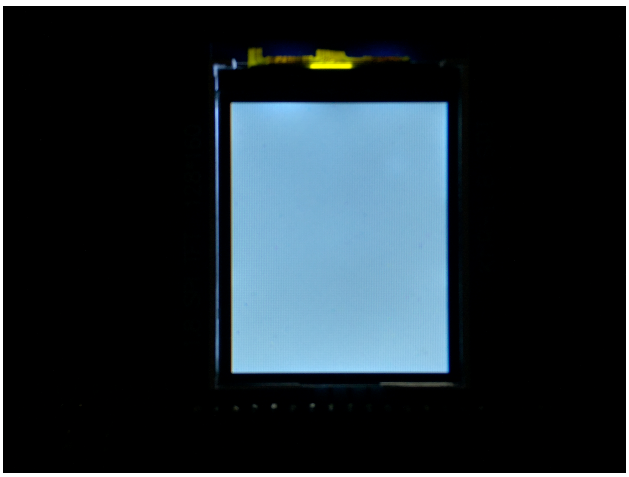

Figure 10. The LCD Screen upon booting up.

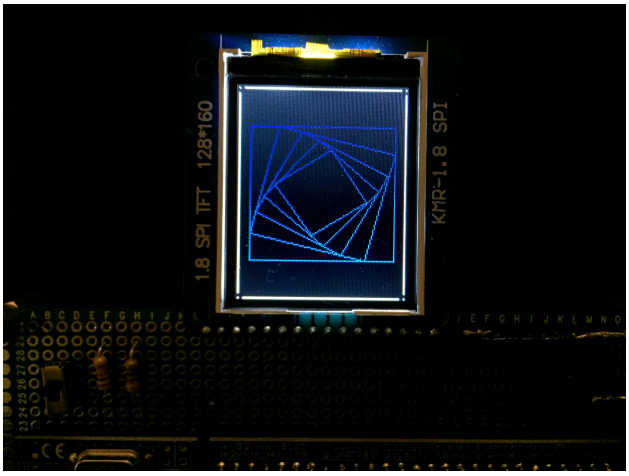

Figure 11. The LCD screen displaying the five rotating blue squares.

Testing the functionality of the IR receiver required the use of an oscilloscope. The positive lead of the oscilloscope probe was attached to the "out" pin of the IR receiver. The ground lead of the probe was simply attached to a ground source. When sending a signal to the receiver, a square-like waveform appeared on the screen as shown in figure 12.

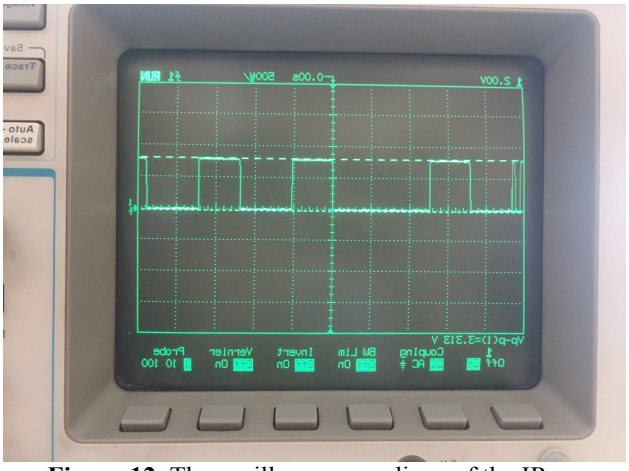

Figure 12. The oscilloscope readings of the IR receiver.

The IR receiver and LCD screen interaction test ran successfully. As the code ran, the sensor was able to receive a signal from the TV remote controller. Pressing any button on the remote control triggers the code. When a signal is received, the console prints a statement saying that a signal was received. The LCD's image and the console output can be found in figures 13 to 15.

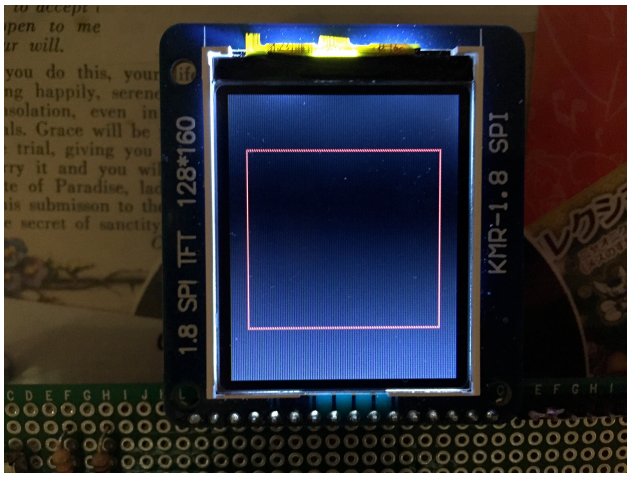

Figure 13. The LCD screen displaying the red square.

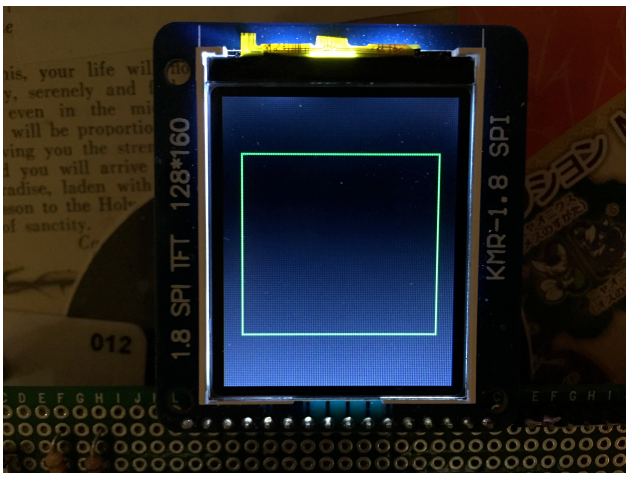

Figure 14. The LCD screen displaying the green square.

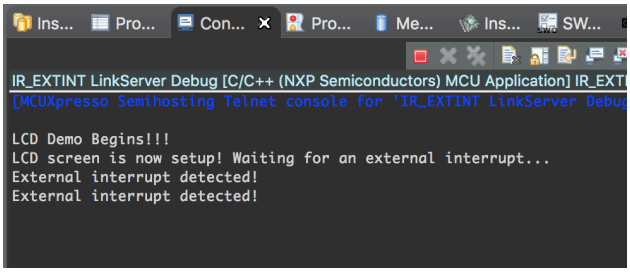

Figure 15. The console printing a message that an IR signal has been received.

5. Conclusion

The main objectives for this lab were to test the LCD screen and to have an IR receiver interact with it. The objectives were all completed successfully. The LCD screen along with the testing code was able to display a set of rotating squares using the appropriate equation. The IR receiver was also able to detect a signal from a TV remote control, and it was able to interact with the screen. The lab was a great way to show off the capabilities of the LCD screen and the IR receiver. It was also a solid way of demonstrating the interactive capabilities of the IR receiver along with the screen. Overall, the lab was a great success.

6. Acknowledgement

Thank you San Jose State University's Software & Engineering Society (SCE) for providing the necessary tools and equipment (solder, soldering iron, parts kit, etc) to complete this prototype board.

7. References

[1] LPCXpresso 1769 CPU Module Datasheet
[2] 1.8 SPI TFT 128*160 LCD Screen Datasheet
[3] TSOP532 IR Receiver Datasheet
[4] H. Li, GitHub Files

8. Appendix

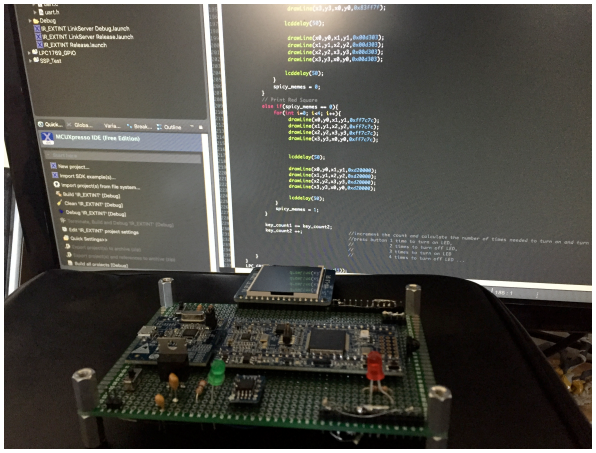

Figure 16. The prototype board next to the MCUXpresso Code.

Figure 17. The prototype board connected to the computer.

LCD Screen Rotating Square Test Main Source Code:

```
/*
===============================================================
==============
 Name        : DrawLine.c
 Author      : $RJ
 Version     :
 Copyright   : $(copyright)
 Description : main definition
===============================================================
==============
*/

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include  "LPC17xx.h"                                   /* LPC17xx
definitions */
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/* Be careful with the port number and location number, because

some of the location may not exist in that port. */

#define PORT_NUM            0


uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];


#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29



#define swap(x, y) {x = x + y; y = x - y; x = x - y ;}

// defining color values

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF



int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;


void spiwrite(uint8_t c)

{

 int pnum = 0;

 src_addr[0] = c;

 SSP_SSELToggle( pnum, 0 );

 SSPSend( pnum, (uint8_t *)src_addr, 1 );

 SSP_SSELToggle( pnum, 1 );

}



void writecommand(uint8_t c)

{

 LPC_GPIO0->FIOCLR |= (0x1<<21);

 spiwrite(c);

}



void writedata(uint8_t c)

{

 LPC_GPIO0->FIOSET |= (0x1<<21);
```

```
 spiwrite(c);
}

void writeword(uint16_t c)
{
 uint8_t d;
 d = c >> 8;
 writedata(d);
 d = c & 0xFF;
 writedata(d);
}

void write888(uint32_t color, uint32_t repeat)
{
 uint8_t red, green, blue;
 int i;
 red = (color >> 16);
 green = (color >> 8) & 0xFF;
 blue = color & 0xFF;
 for (i = 0; i< repeat; i++) {
  writedata(red);
  writedata(green);
  writedata(blue);
 }
}

void  setAddrWindow(uint16_t  x0,  uint16_t  y0,  uint16_t  x1,
uint16_t y1)
{
 writecommand(ST7735_CASET);
 writeword(x0);
 writeword(x1);
 writecommand(ST7735_RASET);
 writeword(y0);
 writeword(y1);
}

void  fillrect(int16_t  x0,  int16_t  y0,  int16_t  x1,  int16_t  y1,
uint32_t color)
{
 int16_t i;
 int16_t width, height;
 width = x1-x0+1;
 height = y1-y0+1;
 setAddrWindow(x0,y0,x1,y1);
 writecommand(ST7735_RAMWR);
 write888(color,width*height);
}

void lcddelay(int ms)
{
 int count = 24000;
 int i;
 for ( i = count*ms; i--; i > 0);
}

void lcd_init()
{
 int i;
 printf("LCD Demo Begins!!!\n");
 // Set pins P0.16, P0.21, P0.22 as output
 LPC_GPIO0->FIODIR |= (0x1<<16);
 LPC_GPIO0->FIODIR |= (0x1<<21);
 LPC_GPIO0->FIODIR |= (0x1<<22);
 // Hardware Reset Sequence
 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);
 LPC_GPIO0->FIOCLR |= (0x1<<22);
 lcddelay(500);
 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);
 // initialize buffers
 for ( i = 0; i < SSP_BUFSIZE; i++ )
 {
   src_addr[i] = 0;
   dest_addr[i] = 0;
 }
 // Take LCD display out of sleep mode
 writecommand(ST7735_SLPOUT);
 lcddelay(200);
 // Turn LCD display on
 writecommand(ST7735_DISPON);
 lcddelay(200);
}

void drawPixel(int16_t x, int16_t y, uint32_t color)
{
 if ((x < 0) || (x >= _width) || (y < 0) || (y >= _height))
```

```
 return;

 setAddrWindow(x, y, x + 1, y + 1);

 writecommand(ST7735_RAMWR);

 write888(color, 1);

}



/*************************************************************
**************

** Descriptions:        Draw line function

**

**  parameters:                  Starting  point  (x0,y0),  Ending
point(x1,y1) and color

** Returned value:        None

**

*************************************************************
*************/


void  drawLine(int16_t  x0,  int16_t  y0,  int16_t  x1,  int16_t  y1,
uint32_t color)

{

 int16_t slope = abs(y1 - y0) > abs(x1 - x0);

 if (slope) {

  swap(x0, y0);

  swap(x1, y1);

 }

 if (x0 > x1) {

  swap(x0, x1);

  swap(y0, y1);

 }

 int16_t dx, dy;

 dx = x1 - x0;

 dy = abs(y1 - y0);

 int16_t err = dx / 2;

 int16_t ystep;

 if (y0 < y1) {

  ystep = 1;

 }

 else {

  ystep = -1;

 }

 for (; x0 <= x1; x0++) {

  if (slope) {

   drawPixel(y0, x0, color);

  }

  else {

   drawPixel(x0, y0, color);

  }

  err -= dy;

  if (err < 0) {

   y0 += ystep;

   err += dx;

  }

 }

}


/*

 Main Function main()

*/



int main (void)

{

           uint32_t pnum = PORT_NUM;

           pnum = 0 ;

           if ( pnum == 0 )
                     SSP0Init();

           else
                     puts("Port number is not correct");

           lcd_init();

           while(1){
                     lcddelay(100);

                     fillrect(0,        0,        ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, BLACK);

                     lcddelay(100);

                     drawPixel(2,2,WHITE);
                     drawPixel(2,158,WHITE);
                     drawPixel(126,2,WHITE);
                     drawPixel(126,158,WHITE);

                     drawLine(2,6,2,154,WHITE);
                     drawLine(126,6,126,154,WHITE);

                     drawLine(6,2,122,2,WHITE);
                     drawLine(6,158,122,158,WHITE);

                     int x0,x1,x2,x3,y0,y1,y2,y3;

                     x0 = 10;
                     x1 = 118;
                     x2 = 118;
                     x3 = 10;

                     y0 = 30;
                     y1 = 30;
                     y2 = 130;
                     y3 = 130;
```

```
                    int i = 0;
                    int temp = 0;

                    lcddelay(100);

                    for(i=0; i<5; i++){
                              // drawLine Functions
                              drawLine(x0,y0,x1,y1,0x1d76c4);
                              drawLine(x1,y1,x2,y2,0x1d76c4);
                              drawLine(x2,y2,x3,y3,0x1d76c4);
                              drawLine(x3,y3,x0,y0,0x1d76c4);

                              // Set new X values
                              temp = x0;
                              x0 = x0+(0.2*(x1-x0));
                              x1 = x1+(0.2*(x2-x1));
                              x2 = x2+(0.2*(x3-x2));
                              x3 = x3+(0.2*(temp-x3));

                              // Set new Y values
                              temp = y0;
                              y0 = y0+(0.2*(y1-y0));
                              y1 = y1+(0.2*(y2-y1));
                              y2 = y2+(0.2*(y3-y2));
                              y3 = y3+(0.2*(temp-y3));

                              //  Set  a  delay  so  the  squares
appear one at a time
                              lcddelay(100);
                    }
                    break;
          }

          printf("Done!");

          return 0;

}
```

External Interrupt Test Main Source Code:

```
/*
===============================================================
==============
 Name        : IR_EXTINT.c
 Author      : $(author)
 Version     :
 Copyright   : $(copyright)
 Description : main definition
===============================================================
==============
*/

#ifdef __USE_CMSIS
#include "LPC17xx.h"
#include "timer.h"
#include "uart.h"
#include "extint.h"
#include "type.h"
#include <time.h>
#endif

#include <cr_section_macros.h>

#include <stdio.h>

// TODO: insert other include files here

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "LPC17xx.h"                                  /* LPC17xx
definitions */
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/* Be careful with the port number and location number, because

some of the location may not exist in that port. */

#define PORT_NUM            0

uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];

#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29

#define swap(x, y) {x = x + y; y = x - y; x = x - y ;}

// defining color values

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF

int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;

void spiwrite(uint8_t c)

{

 int pnum = 0;

 src_addr[0] = c;

 SSP_SSELToggle( pnum, 0 );

 SSPSend( pnum, (uint8_t *)src_addr, 1 );

 SSP_SSELToggle( pnum, 1 );

}

void writecommand(uint8_t c)

{

 LPC_GPIO0->FIOCLR |= (0x1<<21);

 spiwrite(c);

}

void writedata(uint8_t c)

{

 LPC_GPIO0->FIOSET |= (0x1<<21);

 spiwrite(c);

}

void writeword(uint16_t c)

{

 uint8_t d;

 d = c >> 8;
```

```
 writedata(d);

 d = c & 0xFF;

 writedata(d);

}

void write888(uint32_t color, uint32_t repeat)

{

 uint8_t red, green, blue;

 int i;

 red = (color >> 16);

 green = (color >> 8) & 0xFF;

 blue = color & 0xFF;

 for (i = 0; i< repeat; i++) {

  writedata(red);

  writedata(green);

  writedata(blue);

 }

}

void  setAddrWindow(uint16_t  x0,  uint16_t  y0,  uint16_t  x1,
uint16_t y1)

{

 writecommand(ST7735_CASET);

 writeword(x0);

 writeword(x1);

 writecommand(ST7735_RASET);

 writeword(y0);

 writeword(y1);

}

void fillrect(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
uint32_t color)

{

 int16_t i;

 int16_t width, height;

 width = x1-x0+1;

 height = y1-y0+1;

 setAddrWindow(x0,y0,x1,y1);

 writecommand(ST7735_RAMWR);

 write888(color,width*height);

}

void lcddelay(int ms)

{

 int count = 24000;

 int i;

 for ( i = count*ms; i--; i > 0);

}

void lcd_init()

{

 int i;
 printf("LCD Demo Begins!!!\n");
 // Set pins P0.16, P0.21, P0.22 as output
 LPC_GPIO0->FIODIR |= (0x1<<16);

 LPC_GPIO0->FIODIR |= (0x1<<21);

 LPC_GPIO0->FIODIR |= (0x1<<22);

 // Hardware Reset Sequence
 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOCLR |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);

 // initialize buffers
 for ( i = 0; i < SSP_BUFSIZE; i++ )
 {

   src_addr[i] = 0;
   dest_addr[i] = 0;
 }

 // Take LCD display out of sleep mode
 writecommand(ST7735_SLPOUT);
 lcddelay(200);

 // Turn LCD display on
 writecommand(ST7735_DISPON);
 lcddelay(200);

}

void drawPixel(int16_t x, int16_t y, uint32_t color)

{

 if ((x < 0) || (x >= _width) || (y < 0) || (y >= _height))

 return;

 setAddrWindow(x, y, x + 1, y + 1);

 writecommand(ST7735_RAMWR);

 write888(color, 1);

}

// TODO: insert other definitions and declarations here

extern uint32_t timer0_m0_counter, timer1_m0_counter;
extern uint32_t timer0_m1_counter, timer1_m1_counter;

//
================================================================
================================================================

int main(void){

          LPC_GPIO0->FIODIR |= (1<<3); //set pin 0.23 as output
          LPC_GPIO0->FIODIR &= ~(1<<11); //set pin 2.11 as input
          LPC_GPIO0->FIOCLR |= (1<<3); //clear pin 0.23

          uint32_t pnum = PORT_NUM;
```

```
	pnum = 0 ;

	if ( pnum == 0 ) SSP0Init();

	else puts("Port number is not correct");

	lcd_init();

	lcddelay(100);

	fillrect(0,   0,   ST7735_TFTWIDTH,   ST7735_TFTHEIGHT,
BLACK);

	lcddelay(100);

	puts("LCD screen is now setup! Waiting for an external
interrupt...");

	while(1){
		EINTInit();   //wait for external interrupt
	}

	return 0;
}
```

External Interrupt Test Function Source Code:

```
/*****************************************************************
*************
 *       $Id::  extint.c  5670  2010-11-19  01:33:16Z  usb00423
$
 *   Project: NXP LPC17xx EINT example
 *
 *   Description:
 *     This file contains EINT code example which include EINT
 *       initialization, EINT interrupt handler, and APIs for
EINT.
 *

*****************************************************************
************
 *  Software  that  is  described  herein  is  for  illustrative
purposes only
 *  which  provides  customers  with  programming  information
regarding the
 * products. This software is supplied "AS IS" without any
warranties.
 * NXP Semiconductors assumes no responsibility or liability for
the
 * use of the software, conveys no license or title under any
patent,
 *  copyright,  or  mask  work  right  to  the  product.  NXP
Semiconductors
 * reserves the right to make changes in the software without
 * notification. NXP Semiconductors also make no representation
or
 * warranty  that  such  application  will  be  suitable  for  the
specified
 * use without further testing or modification.
*****************************************************************
************/
#include "lpc17xx.h"
#include "type.h"
#include "extint.h"
#include "timer.h"
#include "uart.h"
#include <time.h>
volatile uint32_t eint0_counter;
extern uint32_t timer0_m0_counter, timer1_m0_counter;
extern uint32_t timer0_m1_counter, timer1_m1_counter;
int key_count1 = 0;
int key_count2 = 1;
int key_count = 0;
int spicy_memes = 0;
int start;

int x0 = 10;
int x1 = 118;
int x2 = 118;
int x3 = 10;

int y0 = 30;
int y1 = 30;
int y2 = 130;
int y3 = 130;

//=====================

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/* Be careful with the port number and location number, because

some of the location may not exist in that port. */

#define PORT_NUM            0

uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];

#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29

#define swap(x, y) {x = x + y; y = x - y; x = x - y ;}

// defining color values

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF

void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
uint32_t color)

{

 int16_t slope = abs(y1 - y0) > abs(x1 - x0);

 if (slope) {

  swap(x0, y0);

  swap(x1, y1);

 }

 if (x0 > x1) {

  swap(x0, x1);

  swap(y0, y1);

 }

 int16_t dx, dy;

 dx = x1 - x0;

 dy = abs(y1 - y0);

 int16_t err = dx / 2;
```

```
 int16_t ystep;

 if (y0 < y1) {

  ystep = 1;

 }

 else {

  ystep = -1;

 }

 for (; x0 <= x1; x0++) {

  if (slope) {

   drawPixel(y0, x0, color);

  }

  else {

   drawPixel(x0, y0, color);

  }

  err -= dy;

  if (err < 0) {

   y0 += ystep;

   err += dx;

  }

 }

}

//======================

/*********************************************************************
**************
** Function name:             EINT0_Handler
**
** Descriptions:              external INT handler
**
** parameters:                          None
** Returned value:            None
**
*********************************************************************
*************/


void EINT1_IRQHandler (void){

          LPC_SC->EXTINT = EINT1;

          //LPC_GPIO0->FIODIR |= (1<<2);
          if(LPC_GPIO2->FIOPIN &(1<<11))
          {
                    puts("External interrupt detected!");
                    key_count ++;  // key_count +1 when receive
external interrupt
              delayMs(0,500);  //delay used as debouncer
              start = 1;
          }

          if(start == 1){
                    if( key_count == (key_count1 + key_count2))
                    {
                              // Print Green Square
                              if(spicy_memes == 1){
                                        for(int i=0; i<4; i++){

drawLine(x0,y0,x1,y1,0x83ff7f);

drawLine(x1,y1,x2,y2,0x83ff7f);

drawLine(x2,y2,x3,y3,0x83ff7f);

drawLine(x3,y3,x0,y0,0x83ff7f);

                                                                  lcddelay(50);


drawLine(x0,y0,x1,y1,0x00d303);

drawLine(x1,y1,x2,y2,0x00d303);

drawLine(x2,y2,x3,y3,0x00d303);

drawLine(x3,y3,x0,y0,0x00d303);

                                                                  lcddelay(50);
                                        }
                                        spicy_memes = 0;
                              }
                              // Print Red Square
                              else if(spicy_memes == 0){
                                        for(int i=0; i<4; i++){

drawLine(x0,y0,x1,y1,0xff7c7c);

drawLine(x1,y1,x2,y2,0xff7c7c);

drawLine(x2,y2,x3,y3,0xff7c7c);

drawLine(x3,y3,x0,y0,0xff7c7c);

                                                                  lcddelay(50);


drawLine(x0,y0,x1,y1,0xd20000);

drawLine(x1,y1,x2,y2,0xd20000);

drawLine(x2,y2,x3,y3,0xd20000);

drawLine(x3,y3,x0,y0,0xd20000);

                                                                  lcddelay(50);
                                        }
                                        spicy_memes = 1;
                              }

                              key_count1 += key_count2;
                              key_count2                    ++;
//increment the count and calculate the number of times needed
to turn on and turn off led

                                        //press button 1 time
to turn on LED,

                                        //                  2
times to turn off LED,

                                        //                  3
times to turn on LED

                                        //                  4
times to turn off LED ...
                    }
          }
    LPC_GPIOINT->IO2IntEnR = ((0x01 <<11));
          LPC_GPIOINT->IO2IntClr = 0xFFFFFFFF;
          LPC_GPIOINT->IO0IntClr = 0xFFFFFFFF;

}
/*********************************************************************
**************
** Function name:             EINTInit
**
** Descriptions:              Initialize external interrupt pin
and
```

```
**
          install interrupt handler
**
** parameters:                          None
** Returned value:            true or false, return false if the
interrupt
**
          handler can't be installed to the VIC table.
**
************************************************************
*************/
uint32_t EINTInit( void )
{
           LPC_PINCON->PINSEL4 &= ~(3 << 22 );   //set P2.11 as
EINT1
           LPC_PINCON->PINSEL4 |= (1 << 22 );
           LPC_PINCON->PINMODE4 = 0;
                              // for making pull-up use 00
           LPC_GPIOINT->IO2IntEnR |= ((0x01 <<11));
                              /* Port2.10 is rising edge. */
           LPC_GPIOINT->IO2IntEnF &= ~((0x01 <<11));
           LPC_SC->EXTMODE = EINT0_EDGE | EINT3_EDGE ;        /*
INT1 edge trigger */
           LPC_SC->EXTPOLAR  |= 0;                    /* INT0 is
falling edge by default */
           NVIC_EnableIRQ(EINT1_IRQn);
           return 0;
}

/************************************************************
**************
**                            End Of File
************************************************************
*************/
```